What is claimed is:

1. A computer having a memory storing computer-executable instructions supporting plural objects and a mutation object, said mutation object comprising a method for mutating any one of said plural objects.

2. The computer of Claim 1 wherein each one of said plural objects comprises:

a V-table, an interface corresponding to plural methods and an implementation of said methods;

a V-table pointer pointing to said interface;

an interface pointer for each of said methods to a corresponding one of said implementations.

3. The computer of Claim 2 wherein said mutation object comprises:

a mutation interface including a method for changing a selected one of said pointers of the selected object.

4. The computer of Claim 2 wherein said interface comprises a Mutate_Object method.

5. The computer of Claim 2 wherein said mutation object comprises:

a V-table pointer to said mutation interface;

a pointer from said mutation interface to an implementation of said method for mutating.

6. The computer of Claim 3 wherein said mutation

object mutates said V-table pointer so as to change the interface of the one object to a new interface corresponding to a new set of methods.

7. The computer of Claim 6 wherein said method of said mutation object is a Mutate_VTable method.

8. The computer of Claim 3 wherein said mutation object mutates said interface pointer to point to a new implementation, so as to change the implementation of a given one of the methods of said one object.

9. The computer 8 wherein said method of said mutation object is a Mutate_Object method.

10. The computer of Claim 1 wherein each one of said plural objects comprises a state register storing a state of said one object, and wherein said method of said mutation object changes the contents of said state register so as to mutate the state of said one object.

11. The computer of Claim 10 wherein said state register stores the value of a pointer of said one object.

12. The computer of Claim 11 wherein said pointer of said one object comprises a VTable pointer.

13. The computer of Claim 11 wherein said pointer of said one object points to an implementation of a method of said one object.

14. The computer of Claim 11 wherein said mutation object comprises a Mutate_Object method.

15. The computer of Claim 1 wherein said mutation object further comprises a synchronization of the mutation of one of said plural objects with threads running in said one object.

16. The computer of Claim 15 wherein said synchronization comprises mutual exclusion.

17. The computer of Claim 16 wherein said mutual exclusion prevents new threads from accessing said one object while other threads running in said object are permitted to finish.

18. The computer of Claim 15 wherein said synchronization comprises transactional synchronization.

19. The computer of Claim 18 wherein said transactional synchronization rolls back the threads currently running in the one object and then permits mutation of the object.

20. The computer of Claim 15 wherein said synchronization comprises swizzling.

21. The computer of Claim 20 wherein said swizzling comprises suspending threads running in said one object, mutating said one object and modifying the states of the suspended in accordance with the mutation of the one object, and thereafter reactivating the suspended threads.

22. The computer of claim 21 wherein thread states are swizzled between clean points in the thread execution, whereby the thread becomes suspended at a clean point

23. The computer of Claim 1 wherein one of said plural objects comprises an interposition object formed by said mutation object mutating a particular one of said plural objects and a copied object at least nearly identical to said one particular object, said interposition object differing from said one particular object in that said one particular object has a pointer to said copied object and a method of interposition between threads seeking said one particular object and said copied object.

24. The computer of Claim 23 wherein said interposition method comprises a filter.

25. The computer of Claim 24 wherein said wherein said filter is a read-only filter.

26. The computer of Claim 24 wherein said filter provides access based upon the identity of the requesting thread.

27. The computer of Claim 23 wherein said copied object is a copy of the one particular object.

28. The computer of Claim 27 wherein said interposition object is formed by copying said one particular object and mutating the resulting copy while said copied object is said one particular object.

29. The computer of Claim 7 wherein said new implementation corresponds to a software upgrade.

30. The computer of Claim 7 wherein said new implementation is a higher speed I/O driver.

31. The computer of Claim 7 wherein said new implementation comprises recently loaded code.

32. The computer of Claim 8 wherein said new implementation comprises a different arithmetic algorithm.

33. The computer of claim 8 where said new implementation is a version of an algorithm where specific conditions are assumed to be true, where the version is mutated back to a version when the conditions are no longer true.

34. The computer of claim 33 wherein some of the parameters of the method are assumed to be constant.

35. The computer of claim 34 where the version is generated by a compiler through constant folding.

36. The computer of claim 33 where specific assumptions are made of the objects the method accesses.

37. The computer of claim 36 where the assumption is the location of an object.

38. The computer of claim 36 where the assumption is the value of a field of the state of the object.

39. The computer of claim 36 where the said version is generated through constant folding.

40. The computer of claim 36 where the said version is generated through inlining.

41. A computer operating system capable of supporting plural objects running in a computer having a working memory, said computer operating system comprising:

a kernel resident in said working memory at link time; and

a loadable mutation object resident at link time outside of said working memory and dynamically loadable into said working memory at run time upon demand of one of said application programs, said mutation object comprising an interface with methods for mutating any one of said plural objects.

42. The computer of Claim 41 wherein said kernel comprises a loader for loading said mutation object into said working memory in response to a demand from one of said plural objects.

43. The computer of Claim 41 wherein said computer further comprises a storage memory separate from said working memory, said loadable mutation object residing at link time in said storage memory.

44. The computer of Claim 41 wherein said loader loads said mutation object from said storage memory to said working memory.

45. The computer of Claim 41 wherein said loadable mutation object is terminable from said working memory upon lack of demand therefor by said plural objects.

46. The computer of Claim 41 wherein said kernel of said operating system comprises a Namespace for registering said mutation object upon said mutation object being loaded into said working memory, whereby said mutation object

becomes available to each of said plural objects through said Namespace.

47. The computer of Claim 46 wherein said Namespace comprises an object supporting plural interfaces exposable by said plural objects, said plural interfaces comprising:

a query interface, through which one of said plural objects invokes said mutation object;

an add reference interface by which said Namespace manages each request for said mutation object from any of said plural objects; and

a release reference interface, by which Namespace manages termination of said mutation object from said working memory.

48. The computer Claim 47 wherein said loader is responsive to said Namespace in maintaining said mutation object in said working memory whenever said add reference interface has a reference to said mutation object.

49. A computerized system comprising at least one computer and having a memory storing computer-executable instructions supporting:

plural objects;

a first address space and a second address space wherein respective ones of said plural objects reside;

a mutation object, said mutation object comprising a method for mutating any one of said plural objects;

one of the plural objects in said first address space comprising a proxy object having a pointer to a destination one of the plural objects in said second address space, whereby other objects in said first address space can invoke said destination object in said second address space through said proxy object in said first address space;

said proxy object formed by mutating a destination object in said first address space, said destination object formed by mutating a proxy object in said second address space.

50. A method of providing object mobility in a computer having a memory storing computer-executable instructions supporting plural objects and having a first address space and a second address space wherein respective ones of said plural objects reside, one of the plural objects in said first address space comprising a proxy object having a pointer to a destination one of the plural objects in said second address space, whereby other objects in said first address space can invoke said destination object in said second address space through said proxy object in said first address space, said method comprising:

mutating said proxy object in said first address space to become said destination object and mutating said destination object in said second address space to become said proxy object, whereby said proxy and destination objects are mobile between said first and second address spaces.